Génération automatique de règles de corrélation pour la détection d'attaques complexes

Erwan Godefroy, Eric Totel, Frédéric Majorczyk, Michel Hurfin

Génération automatique de règles de corrélation pour la détection d'attaques complexes

Erwan Godefroy (Erwan.Godefroy@supelec.fr)*
Eric Totel (Eric.Totel@supelec.fr)†
Michel Hurfin (Michel.Hurfin@inria.fr)‡
Frédéric Majorczyk (Frederic.Majorczyk@intradef.gouv.fr) §

Résumé : Dans les systèmes d'information répartis, des systèmes de corrélation sont nécessaires pour traiter le grand nombre d'alertes de sécurité élémentaires et y identifier des motifs d'attaques complexes. Cependant, la complexité du système rend difficile l'écriture de règles de corrélation à la fois précises et correctes. Dans ce papier, on se propose de montrer qu'il est possible, à partir d'un arbre d'attaque construit par un expert, de générer automatiquement des règles de corrélation exhaustives qui seraient fastidieuses et difficiles à énumérer à la main. Les règles de corrélation ainsi générées sont étroitement liées aux caractéristiques du système surveillé (topologie, services déployés, etc.). Ce processus de transformation est implémenté dans un prototype générant des règles de corrélation exprimées dans le langage de description d'attaque ADeLe.

Mots Clés : corrélation d'alertes, arbres d'attaque, scénario d'attaque, IDS, langage de corrélation, base de connaissances

1 Introduction

Aujourd'hui, les systèmes d'informations complexes sont composés de plusieurs dizaines ou milliers de machines coopérant les unes avec les autres ainsi qu'avec des entités externes inconnues. De multiples systèmes de détection d'intrusion (IDS) et des sondes y sont généralement déployés et génèrent des notifications (messages, alertes) de manière continue et indépendante. Des systèmes de corrélation ont été conçus pour pouvoir traiter cette masse d'information [Val06]. Ces systèmes exploitent des relations connues entre des éléments apparaissant dans le flux d'information pour réduire le nombre d'alertes remontées. L'un des objectifs de la corrélation est de reconstituer des scénarios d'attaques complexes.

La détection d'une attaque complexe suppose d'avoir au préalable spécifié formellement le scénario d'attaque correspondant. Ce travail préliminaire, réalisé par un expert, consiste en deux phases. Tout d'abord, cet expert adopte le point de vue de l'attaquant. Il détermine les combinaisons d'actions élémentaires permettant de réaliser l'objectif d'attaque. Puis il adopte le point de vue du défenseur. Il identifie alors les événements et alertes pouvant être

∗. Supélec, DGA-MI
†. Supélec
‡. Inria
§. DGA-MI
Ce travail a été partiellement financé dans le cadre du projet PANOPTESEC (FP7 - GA 610416)

générés à l'exécution et qui prouvent qu'une étape de l'attaque est réalisée. Pour cela, il est nécessaire d'avoir une connaissance approfondie du scénario d'attaque ainsi que du système surveillé et en particulier des dispositifs de détection déployés. Par exemple, l'expert doit connaître pour chaque IDS i) les éléments surveillés, ii) les événements détectables, et iii) le format des alertes levées. En pratique, peu d'experts disposent de connaissances si étendues. En conséquence, les règles de corrélation sont souvent incomplètes ou erronées.

Nous présentons ici un processus de création de règles de corrélation en grande partie automatisé. L'architecture de ce générateur doit répondre à deux objectifs. La représentation initiale d'un plan d'attaque doit tout d'abord être aussi générale que possible alors que les règles de corrélation doivent être spécifiques à un environnement. Le but est à la fois de simplifier le travail initial de l'expert et également de faire en sorte que son analyse reste valide lorsque l'environnement évolue. L'expert doit simplement décrire les caractéristiques principales d'une attaque sans avoir à identifier les cibles ou observations potentielles. Ensuite, la transformation du plan d'attaque en règles de corrélation doit être automatique. Dans notre solution, la transformation séquentielle prend progressivement en compte différentes informations sur l'environnement. Ces informations ont été préalablement stockées dans une base de connaissances maintenue par des experts distincts. En particulier, cette base contient des informations sur le système surveillé (architecture, cartographie), sur les éléments de surveillance (IDS, sondes) et sur les attaques possibles. Une modification de l'environnement requiert uniquement une nouvelle exécution de la transformation automatique. Toutes ces données sont utilisées pour identifier automatiquement les cibles potentielles de chaque action et déterminer la manière de détecter ces occurrences d'action. A partir de la spécification d'un ensemble d'actions interdépendantes, le processus détermine les observations pouvant attester de la réalisation de ces actions sur le système. La solution proposée a été implémentée sous la forme d'un prototype.

La section 2 décrit brièvement les similarités entre les arbres d'attaque et les arbres de corrélation. La section 3 résume notre approche pour générer automatiquement des règles de corrélation. La section 4 définit la notion d'arbre d'actions et la section 5 décrit la base de connaissances utilisée pour les étapes de transformation automatiques présentées dans la section 6. Enfin, la section 7 conclut l'article.

2 Travaux connexes

Différents travaux se focalisent sur la modélisation des attaques en adoptant le point de vue d'un attaquant ou celui du défenseur. L'objectif de cette section est de mettre en évidence la proximité de ces travaux malgré les différences de points de vue.

Le point de vue d'un attaquant consiste à exprimer une séquence d'objectifs à réaliser pour atteindre un objectif (i.e., que l'attaque réussisse). Les travaux réalisés dans ce domaine s'appuient souvent sur la construction d'un arbre d'attaque. Cette notion a été introduite par Schneier dans [Sch99]. Un arbre décrit une décomposition itérative d'un objectif d'attaque en sous-objectifs. La racine de l'arbre représente l'objectif global et les autres noeuds sont des sous-objectifs. Généralement, une relation binaire (également appelée opérateur) est associée aux noeuds internes de l'arbre. Ces opérateurs peuvent être des noeuds *AND* ou *OR*. Un noeud *AND* (respectivement un noeud *OR*) est considéré réussi lorsque tous ses noeuds fils (respectivement, au moins un) sont réussis. Un certain nombre d'extensions ont été proposées [cY07, Kha09, WpWm11] et définissent de nou-

velles relations binaires. En particulier, ces travaux définissent tous un nouvel opérateur exprimant la séquence temporelle (cet opérateur est appelé *PAND* pour « priority *AND* », *O-AND* pour « Ordered *AND* » ou encore *SAND* pour « Sequence *AND* » en fonction de l'article). Les sous-objectifs associés aux noeuds fils doivent être réalisés séquentiellement (en pratique, le premier sous-objectif de la séquence est le noeud de gauche et le dernier est le plus à droite).

Les arbres d'attaque sont utilisés principalement dans le domaine de l'analyse de risques. Les objectifs exprimés sont souvent informels et ne décrivent pas les actions réelles qui sont réalisées par un attaquant. Étant donné que ces arbres ne décrivent pas ce qui doit être détecté, il est donc difficile d'utiliser cette technique directement pour la détection d'intrusion. [cY07] introduit la notion d'action de l'attaquant au sein d'un arbre d'attaque amélioré dans lequel chaque objectif élémentaire est associé à une feuille décrivant l'action à réaliser pour atteindre l'objectif.

De notre point de vue, l'approche de [cY07] est une première étape vers la reconnaissance d'un scénario d'attaque complexe. Cependant, un système de détection d'attaques complexes analyse des alertes issues des IDS ou des événements venant des sondes du système. L'arbre d'attaque amélioré ne se préoccupe pas de savoir si les actions sont observables.

Un système de corrélation considère au contraire uniquement le point de vue du défenseur : sachant que les configurations et sondes du système sont connues, il infère l'occurrence d'une attaque lorsqu'il détecte des traces de celle-ci, Il existe deux approches principales pour détecter ces attaques : les approches semi-implicites [CO00] et explicites. La première consiste à définir des pré- et post-conditions portant sur l'état du système pour chaque étape de l'attaque. L'approche explicite consiste à définir la manière dont les actions élémentaires peuvent être observées et dans quel ordre ces actions observables doivent être détectées. Ces caractéristiques sont exprimées dans un langage de corrélation spécifique. En particulier, cette approche est adoptée dans *GnG* [TVM04] ou *Orchids* [GLO08]. Ces deux approches proposent un langage pour définir un automate à états finis capable de reconnaître un scénario d'attaque. En particulier, ces langages sont capables d'exprimer (1) une séquence d'événements (similaire à l'opérateur *SAND* issu des arbres d'attaque), (2) un ensemble d'événements non ordonnés (similaire à l'opérateur *AND*) ou un événement parmi plusieurs possibles (similaire à l'opérateur *OR*).

Ces similarités entre les relations exprimées par un arbre d'attaque et un langage de corrélation semblent permettre la définition d'un processus de transformation permettant d'obtenir des règles de corrélation en utilisant un arbre d'attaque comme point de départ. Ceci demande bien entendu l'identification automatique des événements et alertes résultant des actions de l'attaquant exprimées dans l'arbre d'attaque. Ce processus est l'objet de cet article.

3 De l'arbre d'actions à l'arbre de corrélation

Le processus de génération de règles de corrélation à partir d'un scénario d'attaque peut se diviser en quatre grandes étapes. Trois d'entre elles peuvent être automatisées grâce aux données présentes dans une base de connaissances. Ces étapes sont illustrées sur la figure 1.

La première étape consiste à enrichir un arbre d'attaque en spécifiant les actions néces-

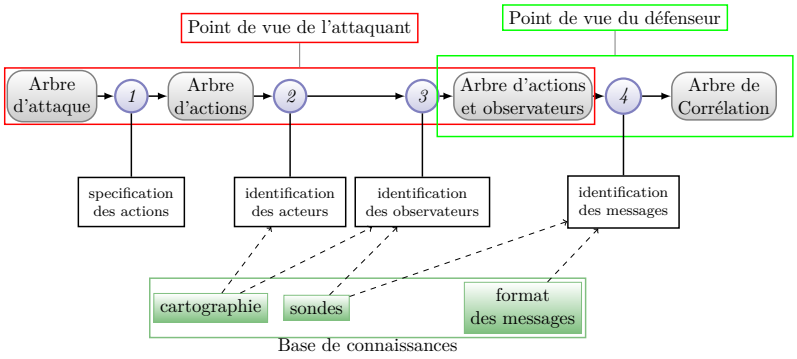

Figure 1: Transformation d'un arbre d'attaque en arbre de corrélation : étapes et structures

saires à la réalisation de chaque sous-objectif. En effet, l'arbre d'attaque initial peut être très générique et décrire un scénario d'attaque connu (comme la propagation de Stuxnet [SK12]) ou bien résulter d'une analyse de risques sur un système particulier. Cependant, cette structure est trop informelle et fait uniquement référence à des sous-objectifs. L'objectif est donc d'atteindre un niveau de granularité suffisant dans la spécification du scénario pour qu'un traitement automatique puisse être effectué. Il est donc nécessaire de spécifier les actions associées à chacune des feuilles de l'arbre d'attaque. Ces actions sont décrites à l'aide d'un langage de description présenté en section 4, permettant de caractériser chaque action par des attributs. Cette étape est réalisée à la main. Toutes les autres sont automatiques et transforment itérativement cette structure arborescente pour construire l'arbre de corrélation final.

La seconde étape permet d'identifier les acteurs (machines, services, processus) du système potentiellement affectés par les actions spécifiées à l'étape précédente. Ces acteurs sont déterminés grâce aux faits renseignés dans la base de connaissances.

La troisième étape vise à identifier les observateurs (sondes, IDS) capables de détecter les actions spécifiées sur les acteurs précédemment identifiés. Cette identification repose à nouveau sur la base de connaissances qui contient des informations sur les capacités de détection des sondes et IDS déployés.

Enfin, la dernière étape consiste à déterminer la syntaxe des messages et alertes potentiellement levés par les observateurs lors de la détection de chacune des actions. Cette structure finale constitue l'arbre de corrélation.

4 Construction de l'arbre d'actions

Dans cette section, on se propose de décrire le processus manuel de construction d'un arbre d'actions à partir d'un arbre d'attaque. L'arbre d'attaque est un outil informel qui n'explicite pas un certain nombre de contraintes dans les scénarios modélisés. Par exemple, il n'est pas nécessaire d'exprimer le fait que deux noeuds de l'arbre font référence à la même machine à partir du moment où il est facile pour un expert de le déduire intuitivement. Ce sont ces informations implicites qui doivent être explicitées dans l'arbre d'actions. Chaque sous-objectif de l'arbre d'attaque doit être traduit en un ensemble d'actions élémentaires

ordonnées avec les opérateurs *AND*, *OR* ou *SAND*. Les actions de l'attaquant sont spécifiées par l'intermédiaire d'un langage de description d'action explicitant l'action et les acteurs (source, cible) qui lui sont liés (cf. Figure 2). Deux éléments identifient une action : son nom et ses attributs (correspondant à la source et à la cible de l'action).

4.1 Convention de nommage

Une action élémentaire d'un arbre d'attaque peut entrer dans deux catégories. Elle peut correspondre à une attaque, ce qui signifie qu'elle appartient à une catégorie connue de classe d'attaque, ou bien elle correspond à une action normale qui ne peut être considérée comme malveillante que dans le cadre du scénario d'attaque complet. Une convention est utilisée pour chacune de ces catégories.

La convention de nommage d'une action élémentaire correspondant à une attaque consiste à utiliser le nom d'une classe d'attaque correspondante issue d'une taxonomie. Cette taxonomie doit lister de manière hiérarchique les différentes classes d'attaques possibles sur un système quelconque. Cette hiérarchie permet d'utiliser le nom d'une classe d'attaque générique incluant potentiellement plusieurs classes d'attaques plus spécialisées. La taxonomie utilisée est CAPEC pour son grand nombre de classes d'attaques référencées.

Les actions normales sont nommées avec les noms d'actions référencées dans la norme CEE . En effet, cette norme a pour objectif de normaliser les logs et présente par conséquent une liste des actions pouvant être trouvées dans des logs. Par exemple, l'initialisation d'une connexion réseau peut être caractérisée par le nom d'action *initiate*.

4.2 Attributs

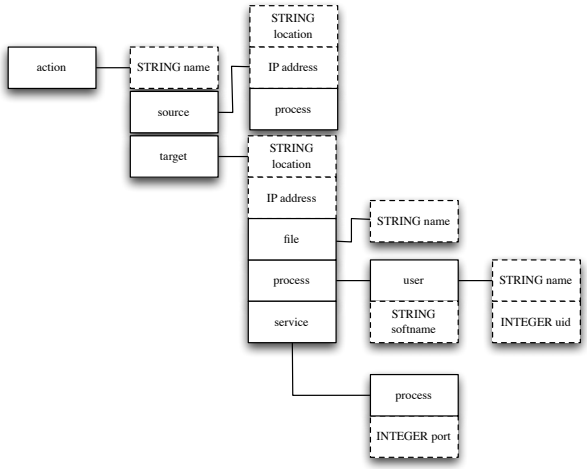

Figure 2: Attributs de description d'action

http ://capec.mitre.org/
http ://cee.mitre.org/

L'objectif final de la description des actions est de permettre de déterminer si l'action est observable dans le système et la manière dont cette observation peut être réalisée. Il est par conséquent nécessaire de savoir si l'action est observable et à quel niveau (réseau, système, applicatif). En pratique, cela revient à déterminer si une action est distante (nécessite le transit de paquets réseaux) ou locale. Cette information est donnée par les deux attributs source et cible (cf. figure 2). Chacun de ces deux attributs qui caractérisent toute l'action est composé de la localisation et de l'adresse IP du noeud sur le réseau. Si ces attributs sont égaux, l'action est locale. De plus, une action a toujours un impact sur une entité du système (i.e., la cible). Cela peut être un fichier, un service en écoute sur un port ou un processus. Un fichier est déterminé par son nom de fichier. Un processus dispose d'un nom et est exécuté avec les privilèges d'un utilisateur (défini par son nom et son uid). Cet ensemble d'attributs est suffisant pour déterminer le niveau du système ciblé par une attaque. Il est important de noter que chaque attribut terminal de la description d'action (en pointillé sur la figure 2) a un type. Si l'un de ces attributs est représenté par une variable, cette dernière hérite du type de l'attribut qu'elle représente.

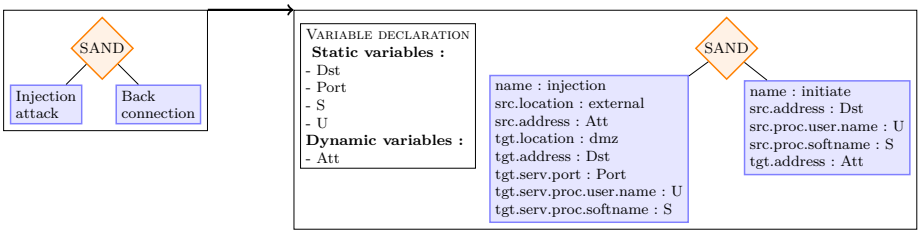

Figure 3: Transformation d'un arbre d'attaque (à gauche) en arbre d'actions (à droite)

Les attributs des actions peuvent appartenir à plusieurs catégories. Ils peuvent être constants ou variables. Un attribut est constant lorsque sa valeur est connue avant la réalisation de l'attaque. Cette constante prend le type de l'attribut qu'elle représente. Un attribut doit être représenté par une variable lorsque sa valeur dépend de l'instance de l'attaque. Pour illustrer ceci, considérons deux attributs particuliers : l'adresse source et cible d'une action. Lors de la spécification d'une action, l'expert peut soit identifier chaque machine (avec son adresse IP) ou bien déclarer une variable. A chaque fois que la même machine est concernée, l'expert utilise la même variable, créant ainsi un lien entre différentes actions. On distingue deux catégories de variables : celles dont toutes les valeurs peuvent être énumérées grâce à la configuration du système décrite dans la base de connaissances et celles dont les valeurs sont indépendantes du système surveillé. La première catégorie est appelée variable statique, la seconde variable dynamique. Par exemple, la cible d'une action peut être une des machines présentes dans un sous-réseau déterminé du système. Les adresses de ces machines peuvent être énumérées, l'attribut correspondant doit être une variable statique. Au contraire, si l'adresse source de l'action appartient à un attaquant externe, l'attribut adresse IP sera une variable dynamique dont la valeur ne pourra être connue qu'en cours d'exécution. On définit la catégorie de chaque variable au moment de leur déclaration. Une variable préalablement déclarée peut être utilisée pour la spécification de plusieurs actions tant qu'elle caractérise des attributs du même type. Par exemple, une variable représentant une adresse IP peut représenter

l'adresse source d'une action puis la cible d'une autre. La figure 3 illustre la transformation d'une petite partie d'un arbre d'attaque en arbre d'actions. Cet arbre d'attaque représente une attaque par injection sur un service d'une machine suivi d'une connexion malveillante depuis cette machine compromise vers la machine de l'attaquant.

5 La base de connaissances

Toutes les transformations automatiques transformant l'arbre d'actions en arbre de corrélation nécessitent une base de connaissances décrivant le système surveillé ainsi que les caractéristiques des sondes déployées. Les modèles tels que [GMHD12] ou M4D4 [MMDD09] apportent une partie des éléments nécessaires. Nous utilisons ici une extension de M4D4. Ce modèle utilise un moteur Prolog permettant l'expression de faits, de règles et de requêtes dans le même langage (le moteur Prolog instancie les variables des requêtes avec les faits disponibles). De plus, ce modèle décrit déjà les aspects topologiques, cartographiques ainsi que les sondes. Notre extension prend en compte les événements et alertes (appelés messages dans ce papier) produits par les sondes et IDS et introduit une taxonomie des attaques possibles et des actions normales dans un système. On se propose de donner les principaux faits de la base de connaissances permettant de spécifier un système à surveiller.

Pour exprimer le fait qu'un serveur d'adresse IP *10.0.0.1* se trouve dans un sous-réseau portant l'identifiant *dmz*, on utilise les faits *node_address(s1, ipv4(10,0,0,1))* et *node_net(s1,dmz)*. Si cette machine héberge un serveur web apache de version *2.2.8* accessible sur le port 80 et exécuté avec les privilèges de l'utilisateur *www-data*, on renseigne le fait *listens(s1, service(process(software(apache, [2,2,8],webserver), www-data), 80))* dans la base de connaissances. Il est également possible de définir un routeur avec le fait *gateway(routeur1)*.

Les sondes et IDS sont définis par leur type, leur visibilité topologique et opérationnelle. Ces notions de visibilité étendent celles introduites dans [MMDD09]. La visibilité topologique fait référence à l'ensemble des acteurs qu'un observateur peut surveiller. Il peut s'agir d'un sous-réseau ou d'une machine particulière. Par exemple, une action locale peut uniquement être surveillée par une sonde locale ou un HIDS, alors qu'une action distante peut être vue par une sonde réseau ou un NIDS. En complément, la visibilité opérationnelle caractérise le type d'action qu'un observateur est capable de détecter pour une configuration donnée. Cette visibilité est liée directement au nom de l'action spécifiée dans la sous-section 4.1. Par exemple, on modélise la présence d'une sonde Snort sur le réseau *dmz* par le prédicat *monitors(snort1,dmz)*. La visibilité opérationnelle de la sonde est donnée par les signatures chargées dans la configuration actuelle. Les faits *sensor_configuration(snort1, config)* et *active(config, s22063)* expriment l'inclusion de la signature *22063* dans la configuration actuelle de la sonde. On exprime la classe d'attaque détectée par cette signature avec le prédicat *detects(s22063, command_delimiters*). Lorsqu'une sonde peut détecter une action sans avoir de base de signatures, on exprime directement le fait que la configuration de la sonde lui permet de détecter cette action.

Enfin, on indique le format de message utilisé par chaque sonde ainsi que les champs présents dans ces messages. Par exemple, si notre sonde Snort est configurée pour générer des messages au format CSV, on renseigne le fait *sensor_format(snort1,snort_csv)*. On indique ensuite les types de champs potentiellement disponibles dans les messages générés par la sonde. Les alertes Snort peuvent par exemple contenir une adresse IP source. Cela

est modélisé par les faits *sensor_data_conf(snort1,dconf)* et *config_available_field(dconf, src.addr)* pour indiquer qu'avec la configuration *dconf*, la sonde génère des messages contenant le champ identifié par *src.addr*.

6 Les transformations automatiques

Dans cette section, on se propose de décrire les transformations réalisées lors des étapes automatiques du processus de transformation de l'arbre d'actions en arbre de corrélation.

6.1 Identification des acteurs

Lors de cette étape, tous les attributs correspondant à des variables statiques sont identifiés en adressant des requêtes à la base de connaissances. Dans un premier temps, la liste de tous les arbres correspondant aux instantiations possibles des variables statiques est construite. Deux arbres de cette liste diffèrent au moins sur une valeur d'attribut. Ensuite, étant donné que tous ces arbres traduisent différents chemins d'attaque d'un même scénario, une seule règle de corrélation est générée. La fusion de ces différents arbres consiste à les lier par un opérateur *OR* en tant que racine commune. Une fois cette étape réalisée, toutes les variables statiques ont été instanciées. L'arbre obtenu conserve une logique similaire à l'arbre d'actions initial mais est désormais spécifique à un environnement donné. A la fin de cette étape, certaines machines (appartenant par exemple aux attaquants externes) ne sont pas identifiées et sont toujours représentées par des variables.

6.2 Identification des observateurs

Durant la seconde étape, la liste des observateurs potentiels de chaque action est associée à cette dernière. Ces observateurs sont ceux dont la visibilité topologique et opérationnelle est compatible avec l'action considérée. Les observateurs sont sélectionnés selon leur visibilité topologique avec le prédicat *can_detect(S,O)* qui est vrai si l'observateur S est capable de voir l'acteur O. Si S est une sonde réseau, ce prédicat utilise la règle *route(Src, Dst, Nets)* qui exprime le fait que les paquets réseau transitent de *Src* à *Dst* en passant par les sous-réseaux intermédiaires contenus dans la liste *Nets*. Cette règle permet de trouver les sondes réseaux situées sur le chemin reliant la source à la cible. Parmi ces observateurs, seuls ceux capables de détecter effectivement l'action concernée sont conservés. Ce filtrage est réalisé avec le prédicat *functional_visibility(S,A)* qui est vrai lorsque l'observateur S a une configuration qui lui permet de détecter l'action A. Si cette action correspond à une attaque (et a donc un nom issu de CAPEC), ce prédicat est vrai si la configuration permet de détecter une sous-classe ou une classe parent de la classe d'attaque spécifiée.

6.3 Identification des messages

L'objectif de cette étape finale est de construire une structure décrivant la structure des messages ou alertes générées par les sondes et IDS lors de la détection de chaque action. Ces messages et alertes sont dans des formats différents et ne contiennent pas nécessairement toutes les informations définies dans l'arbre d'actions. Cette étape vise donc à déterminer les caractéristiques des messages pouvant être levés pour chaque action par les observateurs préalablement identifiés. Chaque noeud est ainsi transformé en un sous-arbre contenant les caractéristiques de ces messages. Dans notre modèle, ces caractéristiques se divisent en deux parties : un nom de format et un ensemble de champs exprimant l'information

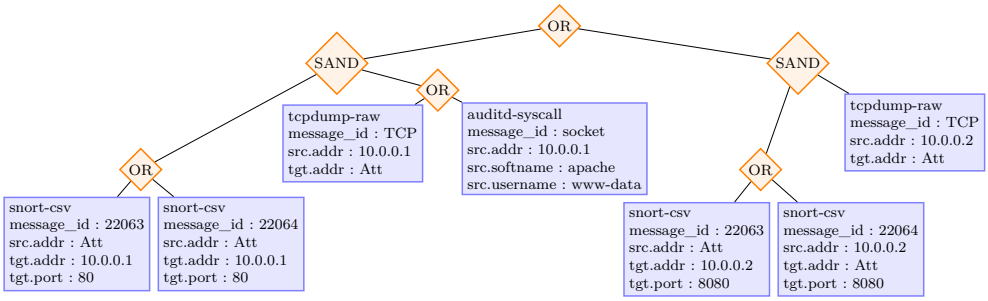

Figure 4: Un arbre de corrélation

contenue dans le message. Le nom du format permet de connaître l'organisation physique des différents champs dans le message réellement généré. Les champs correspondent aux attributs d'actions de l'arbre d'actions. La correspondance est déterminée grâce à un dictionnaire. En plus des champs correspondant aux attributs d'action, un champ nommé *message_id* est ajouté. Ce dernier correspond à l'identifiant de la signature pour une sonde utilisant des signatures et au type du message pour les autres sondes.

Il est possible qu'un observateur puisse observer une action tout en ne fournissant qu'une partie de l'information spécifiée par les attributs de cette dernière. Par exemple, il est peu probable qu'un NIDS fournisse des informations sur le pid d'un processus. Ainsi, seule l'information réellement disponible doit être présente dans les caractéristiques des messages. De plus, il est possible qu'une action puisse correspondre potentiellement à plusieurs messages pour une sonde donnée. Par exemple, SNORT peut fournir plusieurs signatures pour des variantes d'une même classe d'attaque. Dans ce cas, on lie tous les différents messages par un opérateur *OR*, ce qui signifie que l'action est considérée comme détectée si au moins un des messages est présent. De la même manière, lorsque plusieurs observateurs peuvent détecter une même action, les messages produits par chacun d'entre eux sont liés par un opérateur *OR*, ce qui signifie que l'on considére également l'action comme détectée si au moins un des observateurs la détecte. La figure 4 donne un exemple d'arbre de corrélation issu de l'arbre d'actions de la figure 3. Les deux sous-arbres correspondent à deux serveurs WEB (dont les adresses IP respectives sont 10.0.0.1 et 10.0.0.2) pouvant faire l'objet d'attaques. Ils sont surveillés au niveau réseau par l'IDS Snort et une sonde Tcpdump. Une sonde Auditd est placée sur le premier serveur. Les messages de Snort traduisent l'action d'injection alors que les autres messages traduisent la connexion d'un des serveurs vers la machine de l'attaquant.

7 Conclusion

Cet article présente une méthode permettant de générer des règles de corrélation à partir d'un scénario d'attaque et d'une base de connaissances. Ceci est rendu possible par la définition d'un langage de description d'actions permettant de spécifier des actions élémentaires et d'une extension de la base de connaissances M4D4. Cette approche a deux avantages. D'une part les connaissances requises pour créer des règles de corrélation sont divisées en trois domaines faiblement couplés (spécification d'un scénario, cartographie du

système, fonctionnement des sondes). D'autre part, les évolutions du système sont faciles à prendre en compte puisqu'il suffit que la base de connaissances reflète ces évolutions. Le système de génération de règles de corrélation proposé n'est pas uniquement théorique. Cette solution a été implémentée sous la forme d'un système permettant de transformer automatiquement un arbre d'actions en arbre de corrélation puis en règles de corrélation ADeLe en utilisant une extension de M4D4 comme base de faits. Pour quelques attaques complexes spécifiées, nous avons pu produire un ensemble de règles de corrélation précises et complètes qu'un administrateur ne serait pas capable d'écrire à la main (bien que l'environnement considéré était simple).

Références

[CO00] F. Cuppens and R. Ortalo. LAMBDA : A Language to Model a Database for Detection of Attacks. In *Proceedings of the Third Int. Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, number 1907, pages 197–216, 2000.

[cY07] S. Ahmet Çamtepe and B. Yener. Modeling and detection of complex attacks. In *Proceedings of the 3rd Int. Conf. on Security and Privacy in Communications Networks*, 2007.

[GLO08] J. Goubault-Larrecq and J. Olivain. A smell of orchids. In *Runtime Verification*, volume 5289 of *Lecture Notes in Computer Science*. 2008.

[GMHD12] G. Gonzalez Granadillo, Y. Ben Mustapha, N. Hachem, and H. Debar. An ontology-based model for siem environments. In *ICGS3 '11 : 7th Int. Conf. in Global Security, Safety and Sustainability*, volume 99, pages 148–155, 2012.

[Kha09] P. Ahmed Khand. System level security modeling using attack trees. In *Proceedings of the 2nd Int. Conf. on Computer, Control and Communication (IC4)*, pages 1–6, 2009.

[MMDD09] B. Morin, L. Mé, H. Debar, and M. Duccassé. M4d4 : a logical framework to support alert correlation in intrusion detection. *Information Fusion*, 10(4) :285–299, 2009.

[Sch99] B. Schneier. Attack trees : Modeling security threats. *Dr. Dobb's Journal*, 24(12) :21–29, 1999.

[SK12] L. Pietre-Cambacédès S. Kriaa, M. Buissou. Modeling the stuxnet attack with bdmp : Toward more formal risk assessments. In *CRISIS*, 2012.

[TVM04] E. Totel, B. Vivinis, and L. Mé. A Language Driven Intrusion Detection System for Event and Alert Correlation. In *Proceedings ot the 19th IFIP Int. Information Security Conference*, pages 209–224, 2004.

[Val06] F. Valeur. *Real-Time Intrusion Detection Alert Correlation.* PhD thesis, University of California, 2006.

[WpWm11] L. Wen-ping and L. Wei-min. Space based information system security risk evaluation based on improved attack trees. In *Third Int. Conf. on Multimedia Information Networking and Security (MINES)*, page 480–483, 2011.